VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2019 David Bohunovský

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **David Bohunovský**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Craneballs s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Němec, Ph.D.**

Konzultant bakalářské práce: Ing. Matěj Rejnoch

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2019 Bobrovský

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 18. dubna 2019 .....

Na tomto místě bych rád poděkoval kolegům, se kterými jsem na tomto projektu spolupracoval. Dále bych chtěl poděkovat firmě Craneballs za možnost u nich praxi vykonat, a hlavně Ing. Lukášovi Krislovi a Ing. Tomovi Popkovi za cenné rady, které mi byly během praxe předány.

Abstrakt

Bakalářská práce je písemná zpráva o absolvování individuální odborné praxe ve firmě Craneballs s.r.o. Konkrétně se zabývá vývojem hry pro virtuální realitu. Po úvodu je čtenář seznámen s firmou a zadáním hry, které bylo firmou zadáno. Poté následuje popis a konkrétní řešení jednotlivých úkolů, které byly studentem během praxe vykonány. V závěru práce je zhodnocení praxe a shrnutí schopností, které byly během studia a praxe získány.

Klíčová slova: Virtuální realita, unity, uživatelské rozhraní, umělá inteligence, databáze

Abstract

The bachelor thesis is a written report on the completion of individual professional practice in Craneballs s.r.o. Specifically, it deals with the development of virtual reality games. After the introduction, the reader is acquainted with the company and the game that was entered by the company. This is followed by a description and specific solution of the tasks that have been performed by the student during the practice. At the end of the thesis, there is an evaluation of the practice and a summary of the skills acquired during the study and practice.

Key Words: Virtual reality, unity, user interface, artificial intelligence, database

Obsah

Seznam použitých zkratek a symbolů

VR	–	Virtual reality
VRTK	–	Virtual Reality Toolkit
UI	–	User interface
POV	–	Point of View
ID	–	Identity
PK	–	Primary key
FK	–	Foreign key
ORM	–	Object-relational mapping

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Cílem mé bakalářské práce bylo působení ve firmě Craneballs s.r.o [1] (dále jen Craneballs), která se zabývá vývojem her. Ve firmě jsem začal pracovat na začátku 5. semestru jako součást bakalářské praxe. Byl jsem zařazen do týmu Craneballs Academy, který má sloužit k zaučení nových lidí do herního průmyslu. Během praxe jsme jako tým, skládající se ze dvou programátorů a dvou grafiků, dostali za úkol vytvořit hru pro virtuální realitu, konkrétně pro platformu SteamVR [2]. V týmu jsem byl na pozici UI/Frontend vývojáře a také jsem se zabýval umělou inteligencí nepřátel.

Prvně popíšu čím se firma konkrétně zabývá, něco o její historii a také pár slov o týmu, kterého jsem byl součástí. Dále se zaměřím na zadání hry, které nám bylo firmou zadáno, na game desing zmíněné hry a popíši jednotlivé úkoly, které jsem musel při vývoji hry řešit. Nastíním problémy, které nastaly při implementaci a popíší kroky, které jsem podnikl, abych zmíněný problém vyřešil.

Jednotlivé úkoly nejsou seřazeny podle toho jak byly vykonávány, protože jsem se často vracel ke starším věcem, které byly propojeny s novým úkolem, nebo nastal problém, se kterým se dříve nepočítalo.

2 Popis odborného zaměření firmy a zadání projektu

Craneballs je docela mladá firma, která začala roku 2008 s vývojem her na mobilní telefony. Ze začátku pouze na platformu iOS. Největšího rozmachu se firma dočkala v roce 2011 se svou hrou Overkill [3], což byla 2D arkádová střílečka na mobilní telefony. V dnešní době vydává hry také na platformu Android, na kterou můžete najít hry jako Overkill 3 [4], Fling Fighters [6] nebo jejich momentálně nejnovější hru Medieval Smackdown [5]. Studio se také začalo věnovat vývoji her na PC, jejich první hrou ve vývoji je Planet Nomads [7], která je dostupná na platformě Steam.

2.1 Zadání

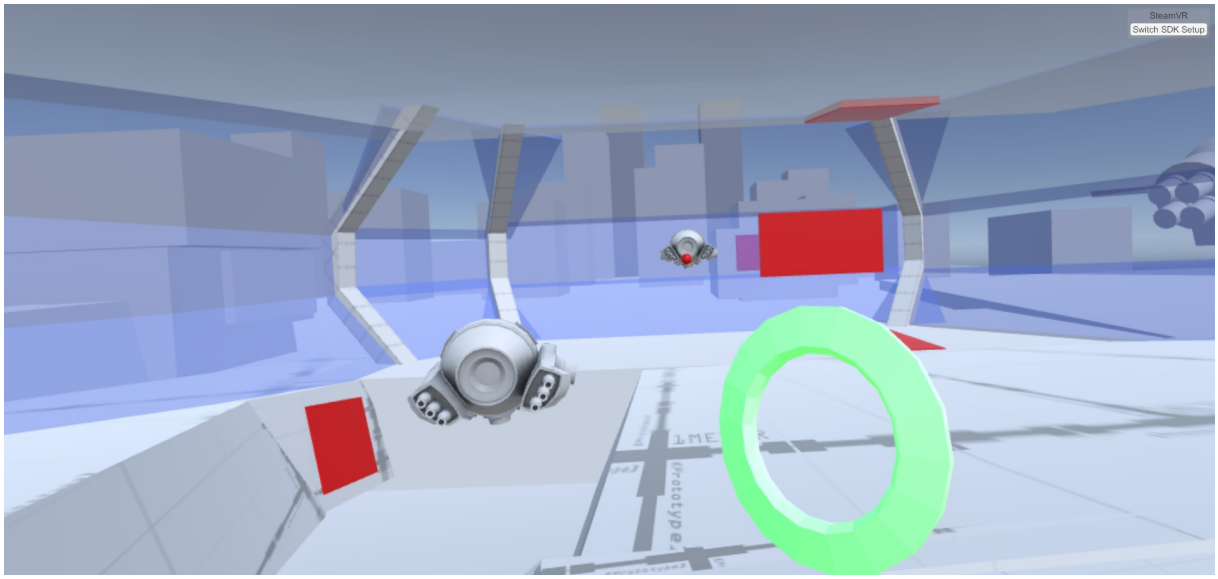
Na začátku jsme jako tým ve firmě obdrželi zadání s hrou, kterou máme za dobu praxe naprogramovat. Jednalo se o hru pro virtuální realitu, která se má hrát způsobem párty módu. To znamená, že jedno zahrání hry trvá pár minut, a na konci se zapíše skóre, které se další hráč může pokusit překonat. Vzhledem je hra zařazena do sci-fi světa. V zadání byly rozepsány jednotlivé iterace, které obsahovaly postup, kterým směrem by se hra měla postupně vyvíjet. V první iteraci bylo například zprovoznit pouze základní scénu. Po přečtení a pochopení zadání jsme si v týmu rozdělili, kdo bude vykonávat jednotlivé úkoly v jednotlivých iteracích. Pro hru měl být použitý herní engine Unity [8], který se používá ve firmě. K práci s virtuální realitou nám byla poskytnuta knihovna VRTK [9].

2.2 Herní design

Hra měla podle zadání vypadat následovně. Po spuštění bude hráč umístěn do herního menu, kde si bude moci hru nastavit a vybrat úroveň, kterou chce hrát. Po spuštění bude hráč umístěn do arény, ve které na něho po nějaké chvíli začnou po vlnách nalétávat nepřátelské drony, kteří budou na hráče útočit. Hráč bude mít v každé ruce disk, podobný jako jste mohli vidět ve filmu Tron: Legacy [10], kterým hra byla inspirována. Disky se budou ovládat pomocí VR ovladačů a to pohybem jako byste chtěli házet v reálném světě. Tyto disky by měl hráč použít k zneškodnění nepřátelských dronů. Hráč se bude moci diskem přímo trefit do drona nebo využít arénu a odrazové plochy, od kterých se disk bude odrážet. Při obraně před nepřátelkými projektily, bude mít hráč 2 možnosti. Buď projektilu uhnout za pomocí menšího úkroku do strany, popřípadě sehnutím, a nebo odražením projektilu za pomocí disku zpět na nepřítele. Hra se dá ukončit dvěma způsoby, buď hráči dojdou životy po tom, co dostane příliš mnoho zásahů, nebo mu vyprší čas, který může v aréně strávit.

Později měly být do hry přidány herní módy, ze kterých si hráč opět bude vybírat před spuštěním úrovně. Konkrétně se mělo jednat o tři, a to Endless, Time Attack a High Ground. High Ground je klasický mód, ve kterém má hráč stanovený počet životů a čas, který může strávit v aréně. Endless, jak už z názvu vypovídá, je mód, který není nějak omezen časem, ale

pouze životy. Jde v něm o to přežít co nejdéle nekonečné útoky nepřátelských dronů, které se časem stávají intenzivnější. V módu Time Attack začíná hráč pouze s krátkým časem, ovšem za každého zabitého nepřítele se mu nějaký čas přičte. Hra opět končí když mu vyprší čas, nebo dojdou životy.



Obrázek 1: Ukázka z rané fáze vývoje

3 Seznam úkolů

- **Seznámení práce s VR** - Na začátku vývoje jsme prvně prozkoumávali co je ve VR možné za pomocí ukázkových scén, které byly obsaženy v knihovně VRTK. Také jsme měli možnost vyzkoušet různé hry na VR.
- **Vytvoření databáze pro hru** - Pro hru musela být vytvořena databáze, která uchovává informace nutné pro běh hry. Vytvoření databáze a otestování trvalo asi 2 dny.
- **Návrh menu** - Jelikož mým primárním účelem byla práce na frontendu/UI hry, musel jsem rozvrhnout menu pro hru. Celá práce na menu trvala přibližně 15 dnů.
- **Uživatelské rozhraní hry** - Bylo potřeba vytvořit smysluplné a přehledné UI, které nebude lidem ve VR motat hlavu. Testování a vybrání správného UI zabralo asi 6 dnů.
- **Umělá inteligence nepřátel** - Nutnost vytvořit logiku nepřítele, jeho pohyb, kdy bude útočit na hráče, kdy se bude pozicovat apod. Kompletní logika nepřítel zabrala kolem 17 dnů.
- **Ladění VRTK** - Narazilo se na hodně problému s knihovnou VRTK, která se chovala jinak, než jsme očekávali a museli jsme zjistil jak tyto problémy opravit.
- **Testování a ladění hry** - Na konci bylo potřeba doladit detaily jako hudbu, zvuky ve hře, a také efekty aby hra nějak vypadala. Detaily jsem se zabýval poslední dny praxe.

4 Využité technologie a programovací metody

4.1 Unity

Pro vývoj jsme používali engine Unity od společnosti Unity Technologies. Je to herní engine určený pro vývoj 2D a 3D her. První verze Unity byla vydaná roku 2005 a byla dostupná pouze pro Mac OS X, navíc bylo možné vytvořenou hru vydat pouze na některé platformy. Cílem bylo vytvořit cenově dostupný herní engine s profesionálními nástroji pro vývoj her. V dnešní době je Unity dostupné i pro Windows a vydávat lze na většinu platforem [8]. Unity je zdarma pro nekomerční použití nebo pokud firma vydělává méně než 100 000 $ [11].

K veškerému vývoji byla použita verze Unity 2018.2.5f1, protože v době vzniku projektu to byla verze nejstabilnější. Během vývoje sice vyšly nové verze, ale neobsahovaly žádnou funkcionalitu, která by byla nutná pro náš vývoj. K psaní skriptů jsme využívali jazyk C# od společnosti Microsoft.

4.2 Knihovna VRTK

Pro práci s VR jsme používali knihovnu VRTK [9] celým názvem Virtual Reality Toolkit. VRTK už v sobě má základní věci pro práci s VR jako například předlohu pro vytvoření headsetu nebo ruk hráče.

Díky VRTK jsme se nemuseli starat o to jak vlastně zajistit aby nám kamery snímaly pohyb rukou nebo hlavy a vůbec komunikaci celého VR hardwaru, což by mohl být projekt sám o sobě.

4.3 Git

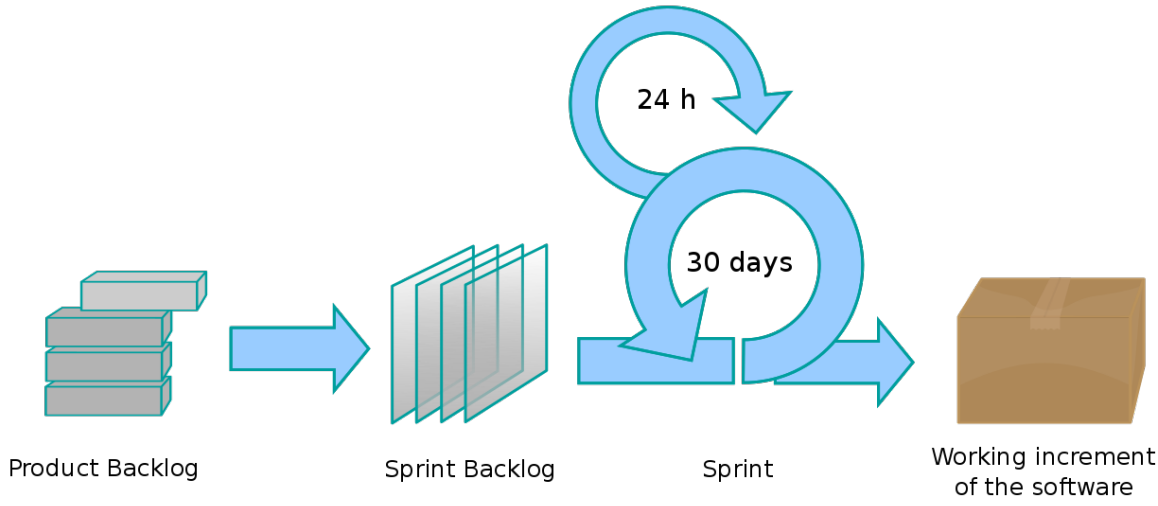
Pro verzování jsme používali verzovací systém Git. Konkrétně Github [12] a k přístupu na git byl využit program Sourcetree [13]. Git je vhodný pokud se na vývoji podílí více lidí. Umožňuje sledovat kdo provádí jaké změny a samotný projekt verzovat, takže když se něco pokazí není problém vrátit projekt do starší verze.

4.4 SQLite

Jelikož databáze byla lokální a nebyla nijak složitá, tak nejvhodnějším kandidátem pro databázový systém byl SQLite [14]. Hlavním důvodem bylo, že SQLite ke svému používání nepotřebuje server, který je nutný u jiných databázových systému, jako například u MySQL [15]. Větší databázové systémy, pravě jako MySQL, mají sice rozšířenější funkcionalitu, ale SQLite obsahuje vše, co je pro naši databázi nutné. Pro úpravy v databázi byl použit program Browser for SQLite [16].

4.5 Scrum

Ve firmě se pracuje podle metody Scrum. Scrum je způsob řízení projektu. Základem Scrumu je sprint, to je doba, většinou 30 dnů, za kterou se má vytvořit funkční část programu. Během sprintu, je na začátku každého dne schůze, kde se rozděluje práce na další den a vyhodnotí se práce z dne předešlého. Špatně vykonané úkoly nebo nevykonané úkoly jsou přehodnoceny a jsou přiřazeny znova [17]. V našem týmu trval jeden sprint obvykle 2 týdny.



Obrázek 2: Metoda Scrum [18]

4.6 Párové programování

Při řešení složitějších úkolů, nebo úkolů, které se prolínaly mezi oběma programátory, jsme využívali párové programování. Spočívá to v tom, že si k jednomu problému sednou dva programátoři a střídají se v psání kódu. Jeden píše a ten druhý ho kontroluje, nebo dává nápady. Správně se mají dvojice střídat, ovšem v našem případě, kdy jsme byli pouze dva, to nebylo možné [19].

5 Konkrétní řešení úkolů

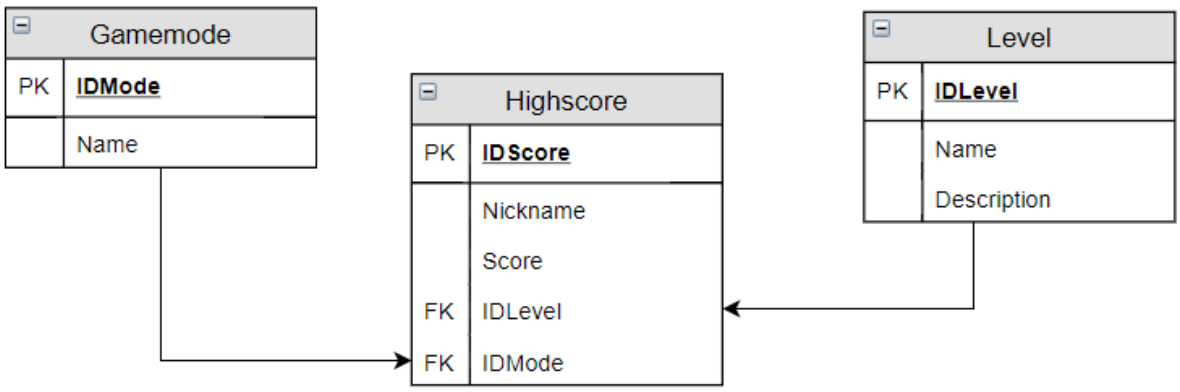
5.1 Seznámení práce s VR

Než jsem se dal k nějakému programování potřeboval jsem se seznámit s virtuální realitou. Každému z nás bylo umožněno si vyzkoušet pár známějších her jako Beat Sabers [20], Gorn [21] nebo Job Simulator [22], abychom dostali obecný přehled o tom jak vlastně VR funguje.

Poté jsem si prohlédl ukázkové scény, které byly připraveny v již zmíněné knihovně VRTK. Díval jsem se hlavně na UI prvky jako tlačítka, posuvníky apod. Jak se jednotlivé prvky ovládají a reagují ve VR. Jestli se něco například neovládá nepohodlně aby nedošlo později k problému kdybych tyto prvky bez váhání začal používat.

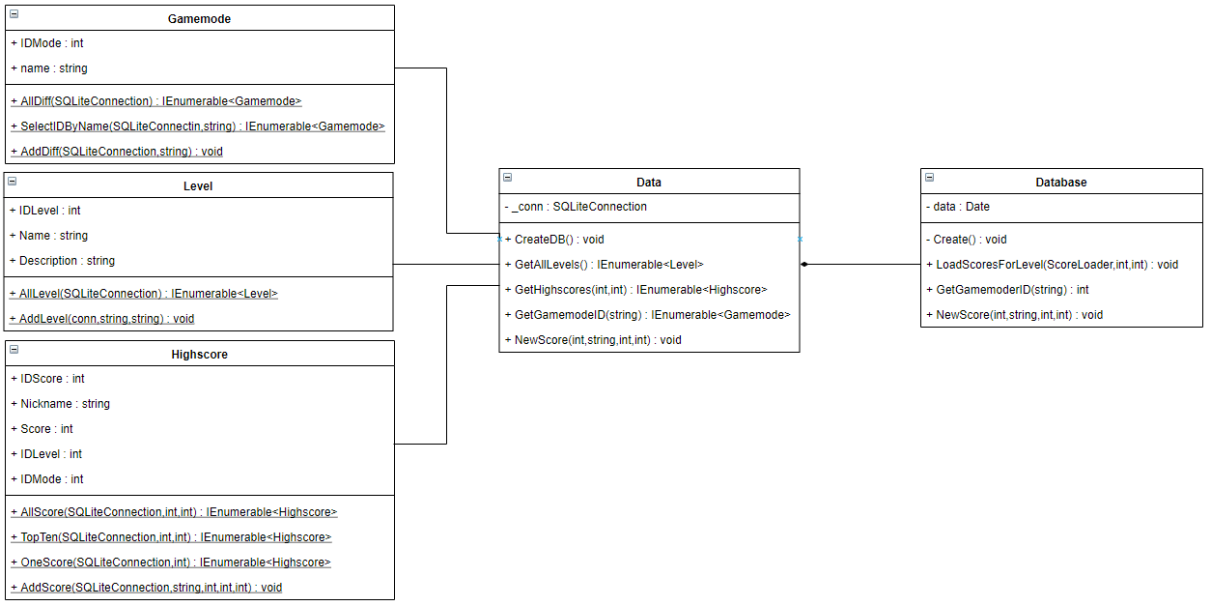
5.2 Vytvoření databáze pro hru

První věcí, kterou jsem se zabýval bylo vytvoření databáze. Jak jsem již zmínil využil jsem k tomu SQLite. Databáze je menší, obsahuje pouze potřebná data nutná pro naší hru. Jedná se o úrovně, které hráč může hrát, obtížnosti, které se vztahují k úrovním a nakonec samostatné skóre, která hráč nahraje.



Obrázek 3: Návrh databáze

Jelikož databáze byla podstatně jednoduchá, rozhodl jsem se pro komunikací s ní využit ORM, které jsem si napsal. Udělal jsem třídu, Gateway, která přímo komunikovala s databází. Následně jsem pro každou tabulku vytvořil třídu, která uchovávala informace a prováděla operace jako výběr nebo zápis do jednotlivých tabulek. A nakonec třídu, se kterou komunikovala aplikace pokud chtěla pracovat s daty z databáze.



Obrázek 4: Třídní diagram

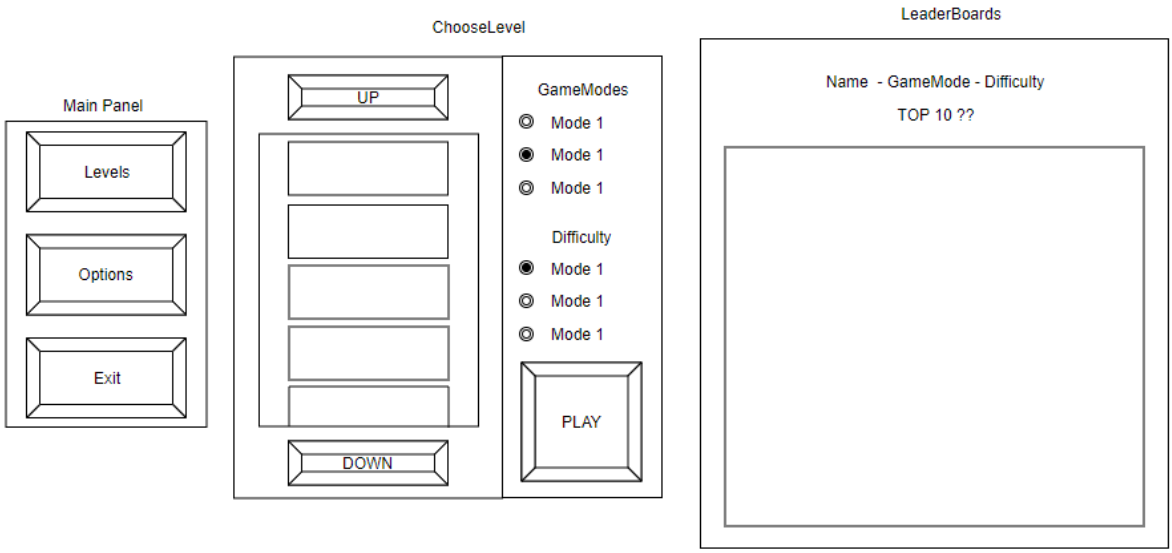
5.3 Návrh menu

Po dokončení databáze jsem se začal věnovat samotnému menu. Prvně jsem si v editoru rozložil, jak bych si výsledné menu představoval, a až poté jsem ho vytvořil v samostatné hře. Po zkoumání jiných VR her jsem zjistil, že jsou 3 způsoby, jakými lze menu udělat.

Menu může být obyčejné, jako to známe u normálních her. Skládá se z tlačítek, posuvníků a zaškrtávacích políček, se kterými uživatel komunikuje a panelů, které zobrazují informace. Rozdíl ve VR je ten, že neklikáme přímo, ale musíme na prvek namířit ukazovátkem, které míří většinou z ovladače. Druhý typ je jedinečný ve VR a to tím, že se skládá z plně interaktivních prvků. Například hráč stojí v prostoru a musí dojít k páčce, za kterou musí osobně zatáhnou, nebo chytnou disketu a vložit jí do počítače apod. Třetí způsob je kombinace mezi normálním a interaktivním menu. Lze vidět například ve hře Blue Effect [23]. Na první pohled vypadá jako normální menu postavené primárně z tlačítek, ale tlačítka jsou interaktivní a musí se stlačit pohybem místo kliknutím.

Plně interaktivní menu nejvíce vystihne, že se hráč opravdu nachází ve virtuální realitě, protože musí někam dojít nebo interagovat s určitým objektem. Při poradě, na které jsme přemýšleli jakým způsobem menu vytvoříme, nás nenapadl žádný způsob ani vzhled, při kterém by plně interaktivní menu mohlo fungovat. Ale napadlo nás jak udělat menu v polo interaktivním stylu. Šlo o to, že hráč už by hru hrál vlastně v menu a pomocí hodu by musel jednotlivé tlačítka v menu trefovat. Místo tlačítka pro start by mohl například sestřelit nějakého drona. Po prozkoumání různých možností, jsem ale došel k názoru, že tímto způsobem by nebylo možné ovládat jiné prvky než tlačítka, po případě zaškrtávací políčka. Což by v případě kdybych chtěl přidat nějaký jiný ovládací prvek bylo velice nepříjemné. Takže jsem nakonec šel variantou jednodu-

chého menu poskládaného z tlačítek, posuvníků a zaškrtávacích políček. Prvotní návrh můžete vidět na následujícím obrázku.



Obrázek 5: Návrh menu

Obrázek 6: Výsledné menu

První okno menu dovolí hráči jít do nastavení, nebo do výběru úrovně. Co se týče nastavení, je zde hlavně nastavení zvuku. Co se týče výběru úrovně, po prvním stisknutí se všechny úrovně načtou z databáze a zobrazí hráči v seznamu. Po vybrání úrovně a herního módu se hráči zobrazí další okno, kde je zobrazeno deset lidí s nejvyšším skóre. Po vybraní kombinace úrovně a módu je umožněno stisknout tlačítko pro start. Po stisknutí je hráč přemístěn do samostatné úrovně.

Po jejím dokončení je hráč přemístěn zpět do menu, kde se mu zobrazí klávesnice pro zadání jména. Po potvrzení je hráčovo skóre zapsáno do databáze a menu je navráceno do stavu před spuštěním úrovně.

Ovšem má práce na menu nebyla pouze skládání prvků do sebe. Některé prvky se chovaly v kombinaci s VR nepřirozeně. Například seznam úrovní skákal sem a tam po zaměření ukazovátkem ve VR, takže bylo skoro nemožné kliknout na úroveň, kterou hráč chtěl. Problém jsem vyřešil tak, že všechny komponenty, které Unity používalo k posouvání listu jsem zakázal a napsal si vlastní skript, který se staral o to, aby se seznam posouval nahoru a dolů. Tento skript jsem namapoval na dvě tlačítka, jedno pro posun nahoru a jedno pro posun dolů.

```
public void OnScroll()
{
    float direction = Down ? -1f : 1f;
    int itemsCount = _scrollRect.content.childCount - (ScrollItemsCount +
        1);
    float offset = (direction / itemsCount) * ScrollItemsCount;
    StartCoroutine(EI_ScrollAnim(offset));
}

public IEnumerator IE_ScrollAnim(float offset)
{
    float startTime = Time.time;
    float startValue = _scrollRect.verticalNormalizedPosition;
    while(true)
    {
        float lerp = Mathf.Clamp01((Time.time - startTime) / EffectDuration)
            ;
        _scrollRect.verticalNormalizedPosition = Mathf.Lerp(0, offset, lerp)
             + startValue;
        if (lerp == 1)
            yield break;
        yield return null;
    }
}
```

Výpis 1: Ukázka skriptu na ovládání posuvníku

Další věcí co jsem potřeboval vyřešit bylo, že tlačítka v menu byla vzhledově upravena tím, že se na ně nahodil materiál, který se skládal z více vrstev. V samotném Unity se dá nastavit, aby tlačítko změnilo vzhled při kliknutí nebo po zamíření na tlačítko, ovšem pouze když se

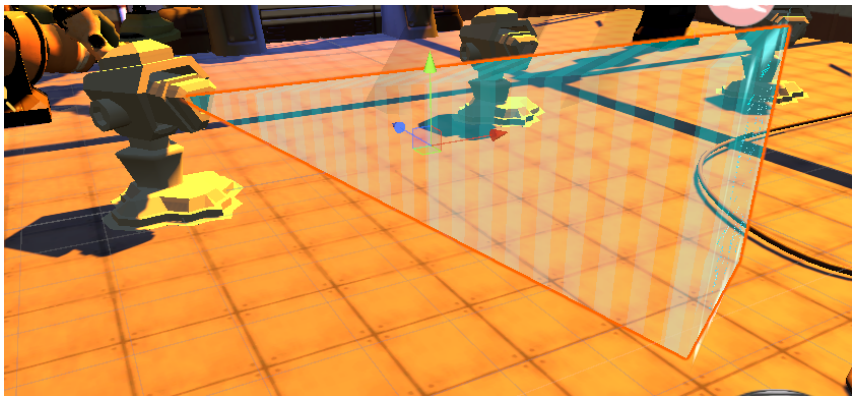
grafika tlačítka skládá pouze z barvy nebo obrázku, neboli spritů. Takže bylo potřeba napsat skript, který dokáže měnit také materiály na tlačítku.

```
private Text _text;
private Image _image;
private Button _button;
private GameConfig _config;
private int _id;

public void OnPointerEnter(PointerEventData eventData)
{
    _text.color = Color.black;
    _image.material = _buttonOn;
}

public void OnPointerExit(PointerEventData eventData)
{
    if (_id != _config.ID.ToString())
    {
        _text.color = Color.white;
        _image.material = _buttonOff;
    }
}

public void SetMaterials(string buttonOn, string buttonOff)
{
    _buttonOff = Resources.Load(buttonOn) as Material;
    _buttonOn = Resources.Load(buttonOff) as Material;
}
```

Výpis 2: Ukázka skriptu na měnění materiálu tlačítek

Poslední věcí, kterou jsem na menu dělal bylo holografické zobrazení. Menu jsem udělat způsobem, že se před hráčem objeví okno, které bude vyřazované z projektoru jako hologram. Po změně okna v menu se první okno zavře a objeví se místo něho jiné. Toto jsem chtěl znázornit animací, při které se hologram zavře a otevře. Paprsek hologramu byl obyčejný čtyřstranný jehlan, na který se aplikoval shader, který vytvářel vzhled hologramu. Pro systém přepínání oken a zavírání hologramů jsem si napsal tři hlavní manažery. Jeden se staral o otvírání samotných oken, ze kterých se skládalo celé menu. Druhý se staral o otvírání a zavírání paprsků hologramu a třetí byl manažer pro funkční tlačítka v menu. Na základě toho, které tlačítko hráč zmačknul,

předával tento manažer ostatním dvěma manažerům informace o tom, jaké okna mají otvírat a které hologramové paprsky k nim mají směřovat. Výsledný vzhled můžete vidět na nasledujícím obrázku.



Obrázek 7: Hologram ve scéně

5.4 Uživatelské rozhraní hry

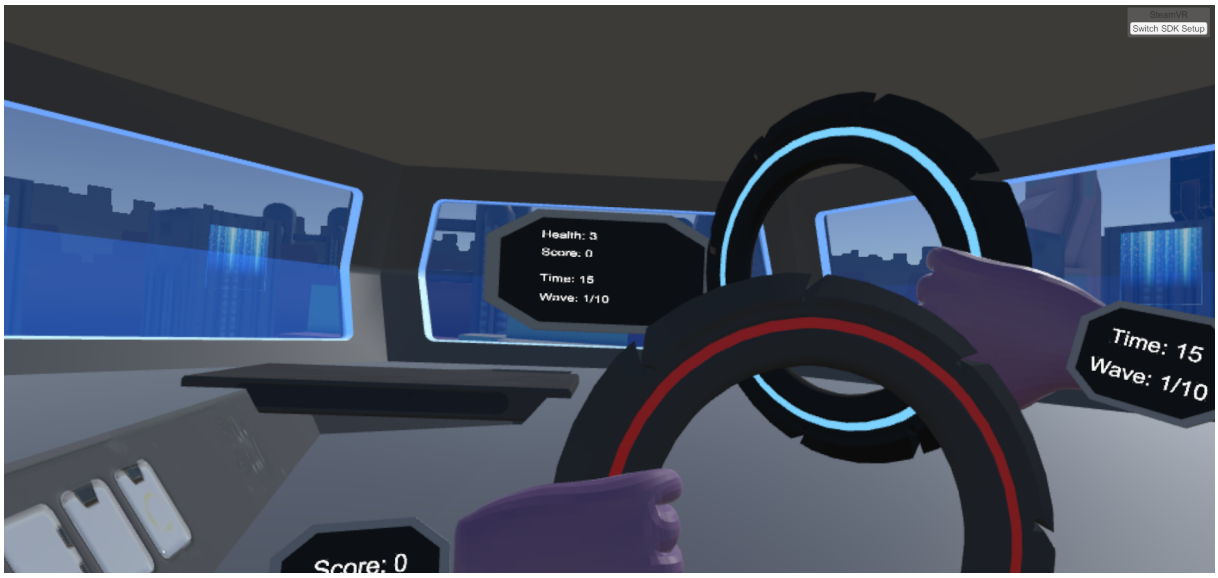
Po dokončení první verze menu bylo mým dalším úkolem vytvořit UI do konkrétní úrovně hry, aby uživatel mohl vidět informace o hře. Informací nebylo moc, primárně šlo o počet životů hráče, zbývající čas do konce, a skóre hráče. Opět se našlo více způsobů, a bylo třeba rozhodnout, který je nejvhodnější.

Prvním způsobem bylo umístit zobrazovací plochy, nebo desky, které by ladily s okolím hry, a na kterých by se zobrazovaly jednotlivé informace. Například v nějaké továrně by to byly monitory, které kdysi zobrazovaly informace o výrobě. Druhý způsob byl umístit informace jakoby na hlavu hráče, nebo jinak řečeno vložit do POV hráče. Takže ať by se hráč díval kamkoliv, měl by informace pokaždé na očích, třeba v levém horním rohu. Třetí způsob byl přidat hráči na ruku něco jako hodinky, na kterých by tyto informace byly zobrazeny.

Při poradě, kde se probíralo UI jsme se shodli, že druhý způsob je špatný v tom, že nemůžeme hráči permanentně nechat uprostřed obrazovky texty s informacemi, takže by se museli dát někam do rohu. Po vyzkoušení jsem se tohoto způsobu vzdal, protože je opravdu nepříjemné skoro šilhat do rohu, jenom proto, abych se podíval na své zbývající životy. Místo toho jsem UI v POV hráče využil na odpočet startu hry, tímto způsobem má hráč stále na očích, kdy začne hra a po startu hry tento text zmizí. Po implementaci ostatních dvou způsobů, jsem tyto způsoby nechal kolegy ve firmě odzkoušet a nechat si dát zpětnou vazbu o tom, co jim na UI vadí, nevadí a co by nechali, popřípadě změnili.

Při variantě s deskami v prostoru byl problém, že když byla deska s informacemi někde bokem, tak se hráč musel otáčet a mezitím dostal zásah od nepřítele. Některým to vůbec nevadilo a přišlo přirozené. Tak stejně to bylo s variantou s hodinkami. První věc byla kam je napozicovat aby hráč nemusel nesmyslně otáčet rukou, když by se chtěl kouknout na své životy nebo čas do

konce. V tom byl trochu problém, protože VRTK vracela pouze pozici ovladače, který hráč drží v ruce, takže jsem musel najít souřadnice, které seděly na pozici hodinek na lidské ruce. Opět se zase nějaké skupině lidí nelíbilo to, že se musí dívat na ruku a tím ztratí na chvíli možnost zaútočit nebo se bránit. Ze začátku se šlo cestou kombinace těchto dvou možností, nakonec po nalezení perfektního místa na tabulku informací jsem se rozhodl hodinky odstranit.



Obrázek 8: Ukázka UI při kombinaci dvou variant

Potřeboval jsem také naprogramovat něco, co se bude o samotné informace starat. Takže nějaký časovač, který bude jak vyplývá z názvu počítat čas a nějaký objekt, který se bude starat o počítání samostatného skóre hráče. Jelikož oba objekty by se měly vyskytovat pouze jednou, rozhodl jsem se pro jejich implementaci využít návrhový vzor Singleton. Singleton slouží k tomu, aby byl objekt v celé scéně pouze jednou a byl k němu globální přístup ze všech ostatních tříd [24]. Tyto oba objekty vyvolávaly události o změně hodnoty, na které reagovaly skripty pro zobrazení a změnily zobrazenou hodnotu.

5.5 Umělá inteligence nepřátel

Další větší částí mé praxe ve firmě bylo programování umělé inteligence nepřátelských dronů. Konkrétně šlo o jejich zrození, pohyb a způsoby útoku nebo léčení svých řad.

Logiku dronů jsem řešil modulárním způsobem. To znamená, že každé chování drona měl na starost jiný modul, který nebyl závislý na ostatních modulech, takže pokud by se jeden modul odebral, dron by fungoval dál akorát neprováděl tu danou funkčnost, kterou měl odebraný modul na starost. Například když má dron moduly pro pohyb, rotaci, vznášení, útok, obranu a odebere se mu modul na útok, tak dron bude fungovat zcela normálně akorát nebude útočit. Pokud nějaký modul potřeboval komunikovat s jiným, řešil jsem to pomocí událostí, na které se daný modul zaregistroval při vytvoření instance.

5.5.1 Rozdělení a zrození nepřátel

Jelikož mělo být více druhů dronů, potřeboval jsem systém, díky kterého budu moci každého nového drona lehce přidat do seznamu možných nepřátel. K tomu jsem využil skriptovací objekty. Ty se používají jako třída, která slouží primárně k ukládání dat [25]. V mém případě jsem měl jeden skriptovací objekt pro každého drona, který obsahoval informace jako jeho ID, počet životů, odměnu za smrt, a jméno šablony, podle kterého se určovalo, jak bude dron vypadat ve hře. Poté objekt samotné úrovně, který obsahoval právě objekty dronů. Na začatku si objekt, který má na starost vytváření dronů, vzal z objektu úrovně seznam těchto dronů. Takže když se přidává nový dron, stačí vytvořit nový objekt s jeho popisem a přidat ho do seznamu. Po načtení seznamu se vytvoří 2 slovníky. Jeden pro samotné vytváření, který obsahuje názvy šablon a samotnou šablonu drona. Podle tohoho se bude dronům přiřazovat vzhled. Druhý slovník se skládá z ID a hodnot odměn za smrt. Po smrti drona dostane hráč správnou odměnu právě na základě výběru z tohoto slovníku. Po načtení těchto potřebných věcí začne objekt, který má na starost zrození nepřátel, vytvářet drony v určitých intervalech do herní arény.

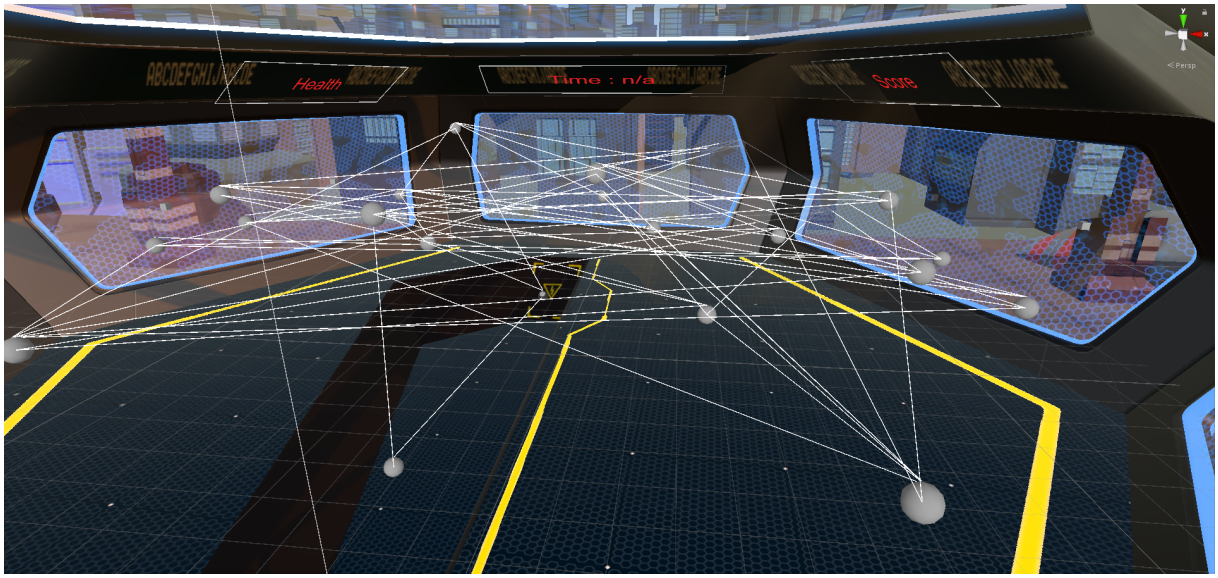
5.5.2 Pohyb nepřátel

U tohoto úkolu jsem strávil asi nejvíce času, protože jsem hodněkrát samotný pohyb nepřátel měnil a přepisoval, než byl ve finální podobě, na které se dá stále ještě pracovat. Popíší tři hlavní způsoby, které jsem zkoušel, jaké u nich nastaly problémy a proč jsem si vybral jeden z nich.

První způsob, se kterým jsem začal byl, že jsem vytvořil síť bodů, po kterých by se drony měly pohybovat. Ve scéně byl objekt, který se choval jako manažer těchto bodů. Každý dron měl agenta, který dronu zadával cestu, která se skládala z několika bodů, po které se měl dron pohybovat. Všechno dohromady to fungovalo následovně.

1. Dron se zrodil a letěl na nejbližší bod.
2. Po dosažení se agent zeptal manažera na novou cestu.
3. Manažer vybral cestu a předal ji agentovi.
4. Agent předal cestu dronu.
5. Dron se vydal po nové cestě.

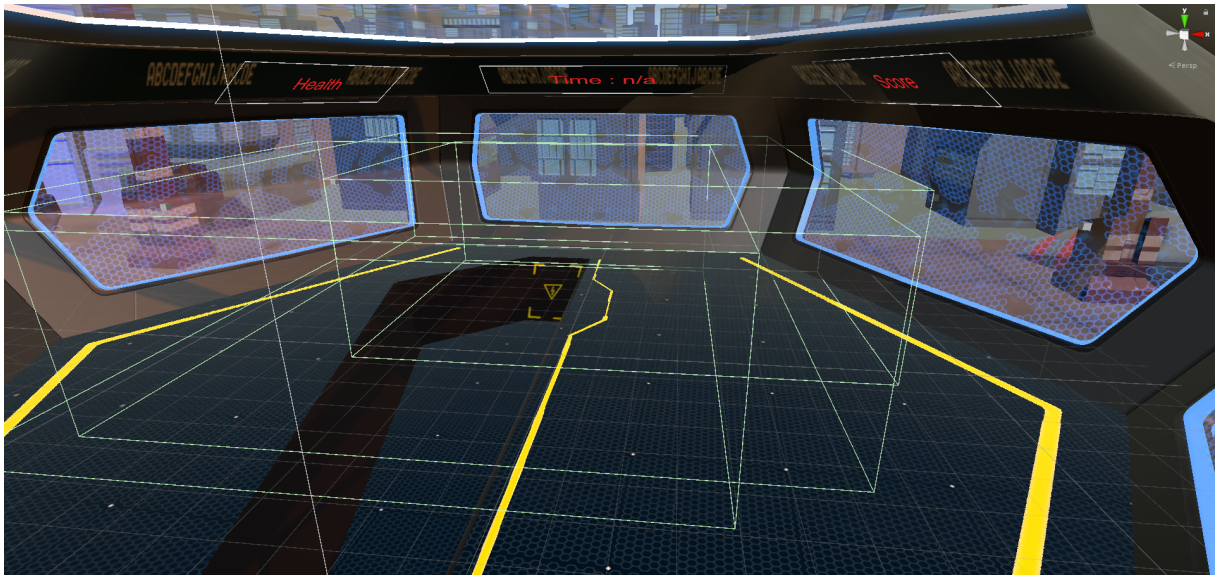
Tento způsob fungoval dobře a měl jsem s ním hodně plánů do budoucna. Ze začátku se cesty vybíraly náhodně, chtěl jsem manažerovi přidat nějaký algoritmus pro hledání cest, přemýšlel jsem o A* [26], a poté mechanismus, který by zabraňoval posílat drona po cestě, na které je už dron jiný. Ovšem po testování a zkoušení, mi po čase přišlo nepřirozené, že jsou drony vždy na určitých místech. Takže aby tato možnost fungovala dobře po dlouhou dobu, musela by být udělána obrovská síť bodů, což v našem případě, kde máme omezený prostor, nebylo vhodné.



Obrázek 9: Waypoint systém

U druhého způsobu jsem to vzal z opačného konce. Místo toho, abych předem udělal nějaké cesty, po kterých se drony budou pohybovat, jsem udělal prostor, ve kterém se budou pohybovat náhodně. Tím zabráním opakovaným pozicím, které vypadaly nepřirozeně. Po mapě bylo rozmístěno více prostorů, ve kterých se drony mohly pohybovat. Při zrození se dronu přiřadil prostor, ve kterém se může libovolně pohybovat. Dron tentokrát neměl žádného agenta, který mu určoval na jaký bod se má vydat, ale sám si hledal náhodné body v přiřazeném prostoru. Když dron našel nový bod, pustil na něho raycast. Raycast je paprsek, který letí od počátku, v mém případě drona, libovolnou vzdálenost směrem, který mu zadáme, v mém případě nový bod a vrátí nám všechny ostatní objekty, kterými prošel. Pokud raycast zaregistroval nějaký jiný objekt, znamenalo to, že by se dron srazil s jiným, takže bylo potřeba najít jiný bod. Zde se objevily problémy. Pokud se dva drony k sobě dostatečné přiblížily, tak se jejich raycasty permanentně dotýkaly a to způsobilo zamrznutí hry. Druhý problém se vyskytl pokud se dron skládal z více objektů, raycast jeho druhou část bral jako jiného drona. Místo podmínkování jestli je to dron samotný nebo jeho část, jsem se rozhodl tento systém trochu změnit.

Tím se dostávám k poslednímu a konečnému způsobu. Každému dronu jsem přidal hledače cest, to je objekt, který se pouze pohybuje po náhodných bodech v prostoru. Samotný dron se pohybuje tím způsobem, že pouze následuje tento objekt. Hledač samotný je 2x rychlejší než dron, takže ho dron nikdy nedožene a tím, že neustále mění směr, má dron i reálnější pohyb. Jelikož hledači cest nezkoumají jestli jim v cestě stojí jiný hledač, tak zmizely problémy při kterých hra zamrzala.



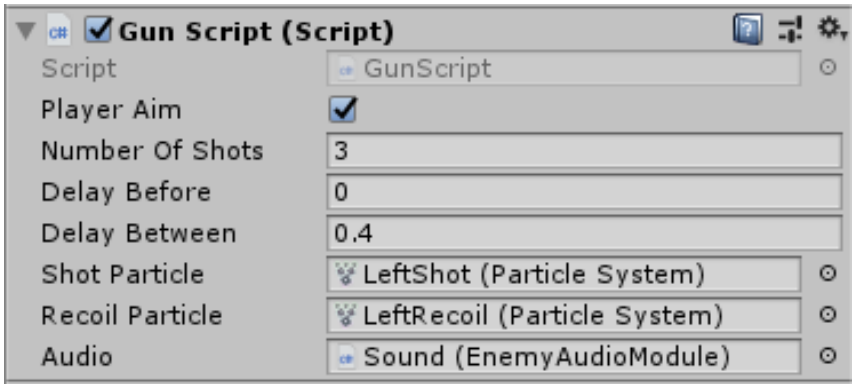
Obrázek 10: Prostorový systém

5.5.3 Útoky nepřátel

Většina dronů útočila na dálku, pro způsob útoku jsem na každého drona přidal prázdné objekty na místa odkud střílel. Tyto prázdne objekty sloužily jako úložiště souřadnic, na kterých se má objevit projektil při výstřelu. Poté jsem udělal útočný modul, který v určitých intervalech na těchto prázdných objektech vytvořil projektily a poslal je směrem k hráči. Problém nastal v době kdy jsem chtěl vytvořit různé druhy útoků. Například jeden dron měl střílet jeden projektil za druhým, zatímco druhý měl vytvořit vlnu projektilů, které hráč musel uhnout nebo se sehnout.

Tento problém jsem vyřešil tím způsobem, že jsem místo úschovny souřadnic přidal na tyto místa opravdové objekty, které se chovaly jako zbraně. Naimplementoval jsem je takovým způsobem, že na nich lze nastavit kolik má zbraň vystřelit projektilů při jednom útoku, interval mezi jednotlivými výstřely a zpoždění výstřelu. Na následujících obrázcích můžete vidět nastavení jednotlivých modulů.

Obrázek 11: Nastavení útočného modulu



Obrázek 12: Nastavení zbraně

Útočný modul poté v určitých intervalech dával zbraním signál, že mají vystřelit. Kolonka Player Aim označovala jestli má zbraň zamířit na hráče nebo má pouze vystřelit před sebe. Toto jsem využíval k již zmíněným různým druhům útoků. Zde přikládám skript, který se stará o vystřelení ze zbraně.

```
public void Shot(Vector3 defaultDirection)
{
    shooting = StartCoroutine(IShot(defaultDirection));
}

private IEnumerator IShot(Vector3 defaultDirection)
{
    yield return new WaitForSeconds(_delayBefore);
    _recoilParticle.Play();
    yield return new WaitForSeconds(0.7f); // Length of recoil anim
    for (int i = 0; i < _numberOfShots; i++)
    {
        GameObject newProjectile = Instantiate(_projectile,_gunTransform.
            position, Quaternion.identity, _parent);
        Rigidbody projectileRigidbody = newProjectile.GetComponent<Rigidbody
            >();
        Vector3 projectileDirection = new Vector3();
        if (_playerAim)
        {
            projectileDirection = (_player.position - _gunTransform.position
                ) + new Vector3(UnityEngine.Random.Range(-0.2f, 0.2f),
                UnityEngine.Random.Range(-0.2f, 0.1f), 0);
        }
        else
        {
            projectileDirection = defaultDirection;
        }
        projectileRigidbody.AddForce(projectileDirection.normalized * 10,
            ForceMode.VelocityChange);
        _shotParticle.Play();
        _audio.PlayShot();
        yield return new WaitForSeconds(_delayBetween);
    }
    ExitCoroutine();
    yield break;
}
```

Výpis 3: Ukázka výstřelu ze zbraně

5.5.4 Lečení a štity nepřítele

Než o léčení, jde spíše o systém štítování. V herním poli je dron, který se místo útočení na hráče soustředí na posílání štítů na ostatní drony. Systém štítování jsem udělal podobně jako střílení. Vytvořil jsem objekt, který se chová jako zbraň, ovšem trochu jinak. Místo projektilu střílí štítovací jednotky. Ty se vytvoří hned, ale nikam se nepřesouvají, zůstávají na zbrani. Poté má dron modul, který se stará o tyto zbraně a zjišťuje, jestli je ve zbrani nabitá jednotka, což znamená, že je připravena ke střelbě. Jakmile má dron vystřelit dá modulu signál, že má vypustit jednu štítovací jednotku. Modul zjistí, jestli je nějaká zbraň nabitá a pokud ano, tak vyšle její štítovací jednotku směrem k jinému dronu, aby se chovala jako jeho štít. Po nějaké době se na místo vystřelené jednotky objeví nová.

Jakmile vystřelená jednotka doletí k určenému dronu, začne kolem něho kroužit. Pokud hráč trefí drona zatímco kolem něho krouží štítovací jednotka, tak se dron nezničí, ale hráčova koule se od něho odrazí a místo drona je zničena štítovací jednotka. Poté má hráč několik sekund na to drona zničit, než je k němu poslána jednotka nová. Pokud hráč zničí drona dříve, než k němu štítovací jednotka dorazí, tak se dron zničí a jelikož štítovací jednotka ztratí svého drona tak se také sama zničí.

5.6 Ladění VRTK

Problémy s VRTK knihovou jsme řešili vždy ve dvou programátorech za využití párového programování. Párkrát se totiž stalo, že knihovna VRTK nefungovala přesně tak, jak bychom chtěli. Když taková situace nastala, potřebovali jsme změnit její chování, a jelikož jsme nechtěli přepisovat přímo originální kód, tak jsme využili dědičnost. Třídu, ve které jsme chtěli chování změnit nebo rozšířit, jsme podědili do naší nově vytvořené třídě a v této třídě jsme upravili chování. Třídu od VRTK jsme následně nahradili za naší.

Mezi takové problémy například patřilo, že při zásahu hráče projektilem občas hráčovi probliklo před očima. Po testování a zkoušení znovu projevení chyby, se zjistilo, že při pohybu hráče se na jeho objektu přepíná collider z triggeru na collider normální. Pokud byl hráč zasažen v momentu, kdy byl aktivní obyčejný collider, tak mezi projektilem a colliderem došlo ke kolizi a to posunulo hráče v herním prostoru do vzduchu. Když tato situace nastala spustil se skript pro teleportaci, aby vrátil hráče zpět na pozici kde stál, a proto mu problikával obraz. Jediný způsob jak toto vyřešit bylo vypnout skript pro teleportaci.

Po vypnutí skriptu nastal jiný problém. Při inicializaci je hráč umístěn na podlahu do určité výšky právě za pomoci tohoto teleportu. Když byl skript vypnut, stávalo se, že se hráč objevil buď mimo arénu nebo ve vzduchu. Tento problém se vyřešil způsobem, že při inicializaci hry se od kamery hráče spustil raycast směrem dolů. Po zasažení prvního objektu, což byla podlaha, se vynutila funkce teleportu, která hráče umístila na tuto podlahu a posléze se skript vypnul aby nedocházelo k problému s problikáváním.

5.7 Testování a ladění hry

V posledních týdnech jsem se věnoval spíše detailům. Jako doladit pohyby dronů, přidat k nim animace, zvuky a efekty. Celkově jsem přidával zvuky a efekty do celé hry. Dále jsem se po celou dobu projektu, ale v této časti nejvíce, věnoval testování samotné hry. Hlavně jsem sledoval, jestli je chování dronů přirozené a celkově hra dělá, to co má.

5.7.1 Animace a efekty

K efektům jsem využil particle systém, který už je obsažený v Unity. Particle systém umožňuje vytvořit a vystřelit poletující částice. Základní nastavení obsahuje vše co je potřeba pro vzhled efektu jako délku trvání, jestli se má efekt opakovat a celkové nastavení toho, jak má efekt vypadat. Ovšem zde opět nebylo obsaženo vše, co bylo potřeba pro nás. Potřeboval jsem, aby se jednotlivé systémy vypínaly, zatímco se druhé zapínaly. Bylo to nutné například k tomu, aby u drona fungovaly levé trysky když letěl směrem doprava, a naopak pravé když letěl doleva. Napsal jsem si k tomu skript, který udržoval stav drona. Jestli letí do leva, do prava nebo stojí na místě a podle daného stavu jsem jednotlivé particle systémy vypínal a zapínal.

Co se týče animací u drona, který vypadá jako kvadrokoptéra, jsem implementoval naklánění, opět podle směru letu. K animacím Unity dodává animátora. Animátor umožňuje nastavit jednotlivé stavy a přechody mezi nimi na základě proměnných, které si zvolíme. Poté můžeme k jednotlivým stavům přiřadit animace, přechody mezi nimi a nastavit, jestli se má animace vyrušit pokud v půlce jedné změní objekt svůj stav. Jelikož jsem chtěl pouze náklánění, což je jeden stav, tak místo použití animátora jsem si na to napsal skript, který nakláněl objekt o určité stupně. Po implementaci se místo naklánění objekt točil o 360 stupňů stále dokola. Po chvilce debugování jsem zjistil, že je to kvůli tomu, že jsem počítal se zápornými úhly. Počítal jsem s nimi, protože jsem chtěl, ať se objekt naklání od nulového úhlu buď o 15 stupňů nad 0 nebo pod 0. Jelikož Unity počítá úhly normálně od 0 do 360 stupňů, potřeboval jsem si stupně přepočítat, abych je měl od -180 do 180. Toho jsem dosáhnul tímto způsobem.

```
var currentAngle = _cachedTransform.localEulerAngles.z;
currentAngle = (currentAngle < 180) ? currentAngle : currentAngle -
    360;
```

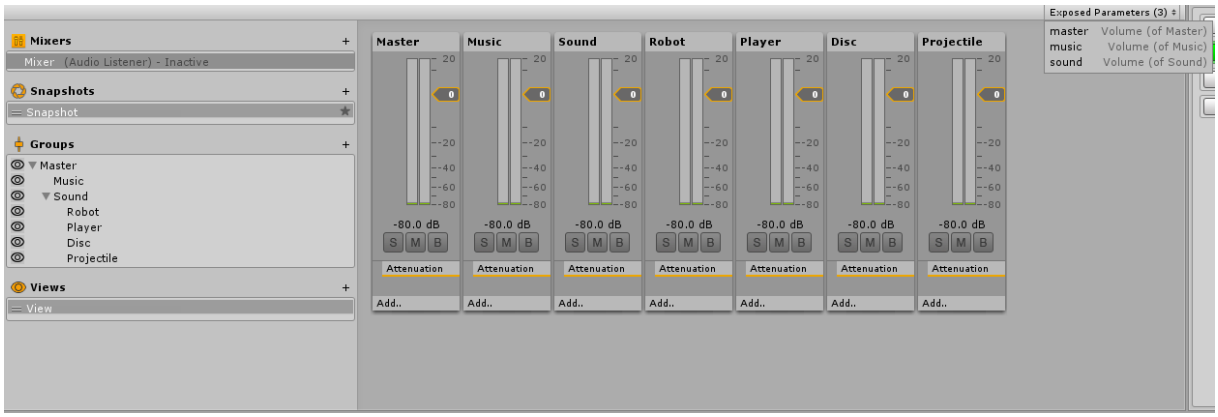
Výpis 4: Ukázka přepočítání úhlu

Problém s efekty nastal, když jsem se snažil využít vlastnosti jednoho shaderu. Shader, který jsme používali na silová pole arény, měl efekt odrazu, když na něj dopadl objekt. Na obyčejných objektech, které byly v základu Unity tento shader fungoval, ovšem na objektech vytvořených grafiky ne. Po zkoušení a upravování vlastností shaderu, jsem se stále nemohl dopracovat k tomu proč efekt nefunguje, tak jsem ve firmě požádal o pomoc. Zjistili jsme, že problém byl v exportu objektů z 3DS Maxu, programu ve kterém pracovali grafici. Po exportu objektu z 3DS Maxu

je totiž objekt otočen o 90 stupňů v ose x, takže shader při výpočtu pozice dopadu nemohl dosáhnout správných souřadnic.

5.7.2 Hudba

Nebudu zde popisovat hodiny, které jsem strávil hledáním zvukových efektů, které byly vhodné pro hru, ale spíše samotnému nastavení hlasitosti. K nastavení hlasitosti jsem použil Audio Mixer, který už je v základu Unity. Dovolil mi nastavit hlasitost zvlášť pro jednotlivé objekty, a tím možnost vytvořit nastavení, kde si uživatel může nastavit zvlášť hlasitost pro zvuky, efekty a hudbu v pozadí.



Obrázek 13: Zvukový mixér v Unity

Na obrázku můžete na levé straně vidět hierarchii zvuků. V nastavení se může nastavit hlasitost Master, Music a Sound. Část, která se nastaví, ovlivní také všechny, které jsou v hierarchii pod ní. Poté se jednotlivé hlasitosti přiřadí k audio prvkům, které ovládají hlasitost na jednolivých objektech. Například k objektu, který pouští hudbu do pozadí, se přiřadí hlasitost Music. Výsledná hlasitost hudby v pozadí poté lze nastavit za pomocí hlasitostí Music a Master.

K samotnému nastavení jsem využil posuvníky, které po změně hodnoty změnily hodnotu v mixéru. Zde byl problém, že posuvník měl hodnoty 1 - 100 zatímco mixér měl hodnoty od +20db do -80db s tím, že nešlo nastavit jako 1 bod na posuvníku se rovná 1 db, protože například 50 procentní hlasitost je asi -6db.

Na webu jednoho herního designera jsem našel článek [27], který se věnoval právě správnému převodu hlasitosti na dB. V článku jsem zjistil že hodnotu posuvníku musím nastavit od 0.0001 do 1 a při převedení na dB použít následující vzorec, s tím že maximální hlasitost je 0 dB. Ve vzorci je X hodnota z posuvníku a Y jsou výsledné decibely.

$$Y = Log_{10}(X) * 20 \tag{1}$$

5.7.3 Testování

Ke konci vývoje jsme s kolegy uspořádali prezentaci pro celou firmu. Sdělili jsme jim, co nám bylo zadáno, čeho jsem dosáhli a jestli by všichni po jednom odkoušeli hru a dali nám nějakou zpětnou vazbu. Tato vazba nám, hlavně mně hodně pomohla k tomu, abych doladil poslední detaily hry. Většině vadilo to, že najednou umřeli a nebyli si vědomi toho, co je zabilo, a ani toho, že jim zbýval poslední život. Na základě této informace jsem trochu předělal UI hry, dal jsem životy přesně před hráče, aby si jich hráč všimnul a přidal jsem effekt těžkého dýchání, když měl hráč poslední život. Další takový detail bylo přepsání času z čistých sekund na minutový systém.

Obrázek 14: Fotografie z testování hry

6 Závěr

Jako tým jsme dosáhli úspěchu a podařilo se nám vytvořit kompletní hru, která dostala název Drone Wars VR. Mně osobně se podařilo splnit všechny úkoly, které mi byly během praxe zadány a dokonce jsem k vývoji přispěl i vlastními nápady, ze kterých se některé zrealizovaly.

6.1 Uplatnění teoretických a praktických znalostí

Během praxe jsem využil znalosti z několika předmětů, které jsem během studia absolvoval. Mezi předměty, které jsem využil patří Programovací jazyky II kvůli znalosti jazyka C#, Uživatelská rozhraní, Úvod do databázových systémů a Databázové a informační systémy. Dále znalost algoritmů a návrhových vzorů z předmětů Algoritmizace I a II, Úvod do softwarového inženýrství a Vývoj informačních systémů. Něco málo zkušeností s vývojem her jsem měl ze samostudia, během kterého jsem si pár her zkoušel naprogramovat.

Ovšem mi také některé znalosti scházely. Například jsem nikdy nepracoval v Unity, ke kterému jsem se dostal až na této praxi. Také jsem vždy na projektech pracoval sám a neverzoval si je, takže mi chyběla znalost práce s Gitem.

6.2 Dosažené výsledky v průběhu praxe

Odbornou praxi v této firmě jsem si zvolil hlavně kvůli tomu, že si myslím, že mi tato praxe otevře dveře do herního průmyslu. Navíc mi bylo předem řečeno, že budu pracovat v týmu, což je taky velmi cenná zkušenost, se kterou jsem se chtěl seznámit.

Za celou praxi jsem získal hodně nových zkušeností, hlavně zmíněnou zkušenost práce v týmu. Jaké to je se domlouvat se zbytkem týmu, na čem kdo bude pracovat, porady o tom, jakým směrem se dál půjde a podobně. Také jsem získal hodně nových praktických znalostí co se týče programování. Celkově bych praxi v této firmě zhodnotil velmi kladně a jsem rád, že mi byla poskytnuta možnost zde pracovat.

Co se týče výsledné hry. Během vývoje jsem zjistil, že ne vše jde podle plánu, který se navrhne na začátku. Konečná podoba hry se víceméně držela herního designu, ale v některých ohledech šla trochu jiným směrem. Hlavní změny byly, že se místo disků přešlo na zbraň, která měla dvě stádia. Buď se chovala jako útočná zbraň, která házela na nepřátelské drony koule, nebo jako štít, který sloužil k odrážení nepřátelských projektilů. Hlavním důvodem bylo, že při testování každý házel disk jiným způsobem, a nešel nastavit tak, aby vyhovoval všem. Další změna byla v nepřátelích, kde místo toho, aby na hráče útočili po vlnách, se objevovali v určitých intervalech. Jedna změna přišla taky na herní módy, kde se v módu Time Attack změnilo trochu chování. Hráči se zrušily životy, byl v tomto módu nesmrtelný, ale za každý zásah se mu ubraly tři vteřiny ze zbývajícího času, takže mód už závisel čistě na čase.

Literatura

[1] Craneballs s.r.o [online]. [cit. 2019-03-10]. Dostupné z: https://www.craneballs.com/

[2] SteamVR [online]. [cit. 2019-03-12]. Dostupné z: https://steamcommunity.com/steamvr

[3] Overkill [online]. Craneballs, 2011 [cit. 2019-04-09].
Dostupné z: https://itunes.apple.com/us/app/overkill/id421659813?mt=8

[4] Overkill 3 [online]. Craneballs, 2014 [cit. 2019-0-09].
Dostupné z: https://play.google.com/store/apps/details?id=com.craneballs.overkill3

[5] Medieval Smackdown [online]. Craneballs [cit. 2019-04-09].
Dostupné z: https://play.google.com/store/apps/details?id=com.craneballs.smackdown

[6] Fling Fighters [online]. Craneballs, 2017 [cit. 2019-04-09].
Dostupné z: https://play.google.com/store/apps/details?id=com.craneballs.flingfighters

[7] Planet Nomads [online]. [cit. 2019-03-12]. Dostupné z: https://www.planet-nomads.com/

[8] HAAS, John. A History of the Unity Game Engine [online]. [cit. 2019-04-10].
Dostupné z: https://web.wpi.edu/Pubs/E-project/Available/E-project-030614-143124/unrestricted/Haas_IQP_Final.pdf.
Faculty of WORCESTER POLYTECHNIC INSTITUTE. Vedoucí práce Brian Moriarty.

[9] VRTK [online]. [cit. 2019-03-12]. Dostupné z: https://vrtoolkit.readme.io/docs

[10] Tron: Legacy [online]. Joseph Kosinski, 2010 [cit. 2019-04-09].
Dostupné z: https://www.imdb.com/title/tt1104001/

[11] Unity Store [online]. [cit. 2019-04-10]. Dostupné z: https://store.unity.com/products/unity-personal

[12] Github [online]. [cit. 2019-03-12]. Dostupné z: https://github.com/

[13] Sourcetree [online]. [cit. 2019-03-12]. Dostupné z: https://www.sourcetreeapp.com/

[14] SQLite [online]. [cit. 2019-03-20]. Dostupné z: https://www.sqlite.org/index.html

[15] MySQL [online]. [cit. 2019-04-10]. Dostupné z: https://www.mysql.com/

[16] SQLite Browser [online]. [cit. 2019-03-20]. Dostupné z: https://sqlitebrowser.org/

[17] SCHWABER, Ken a Jeff SUTHERLAND. The Scrum Guide. 2017.

[18] Scrum [online]. In: . [cit. 2019-04-18].
Dostupné z: https://commons.wikimedia.org/wiki/File:Scrum_process.svg

[19] Párové programování [online]. [cit. 2019-04-10].
Dostupné z: https://www.codementor.io/pair-programming

[20] Beat Saber [online]. Beat Games, 2018 [cit. 2019-04-09]. Dostupné z: http://beatsaber.com/

[21] GORN [online]. Free Lives, 2017 [cit. 2019-04-09].
Dostupné z: https://freelives.net/games/gorn/

[22] Job Simulator [online]. Owlchemy Labs, 2016 [cit. 2019-04-09].
Dostupné z: https://jobsimulatorgame.com/

[23] Blue Effect [online]. DIVR Labs, 2016 [cit. 2019-04-09].
Dostupné z: http://www.blue-effect.com/

[24] SHVETS, Alexander. Dive into Desing Patterns. s. 137-145.

[25] Unity ScriptObject [online]. [cit. 2019-03-12].
Dostupné z: https://docs.unity3d.com/ScriptReference/ScriptableObject.html

[26] Astar [online]. [cit. 2019-03-12]. Dostupné z: http://theory.stanford.edu/ amitp/GameProgramming/AStarComparison.html

[27] FRENCH, John Leonard. The right way to make a volume slider in Unity [online]. 2018 [cit. 2019-04-09]. Dostupné z: https://johnleonardfrench.com/articles/the-right-way-to-make-a-volume-slider-in-unity-using-logarithmic-conversion/